AirJava: Networking for Pervasive Computing

Abstract

Two emerging technologies, pico-cellular wireless and mobile code, promise to provide the networking foundation for pervasive computing. We can envision that within five years vendors will offer portable and embedded devices containing low-cost chips for pico-cellular wireless communications, coupled with virtual machine interpreters that support the exchange of executable programs among networked devices. The combination of these new technologies promises to revolutionize computing and networking, as we know it today. We propose to accelerate this coming revolution by designing and constructing ***AirJava*** adapters that can convert any existing computer-controllable device into a prototype pervasive computing device. Further, we propose to demonstrate ***AirJava*** networking in support of pervasive computing applications. Our ***AirJava*** adapters can be used within pervasive computing test beds at NIST, and can be supplied to other researchers investigating issues associated with pervasive computing. Our ***AirJava*** adapters should also inform commercial developers about design issues surrounding pervasive computing devices. We envision that this work will lead to one or more competency proposals regarding pervasive computing. Of particular future interest could be: (1) security techniques for pico-celluar wireless communication and for mobile code systems, (2) software techniques for specifying interfaces that support dynamic composition of mobile objects, (3) networking protocols for reducing the latency of information access for mobile users, and (4) human-information interaction techniques that adapt to the interface devices at hand.

Background

Industry is developing two, competing wireless transceiver-on-a-chip technologies that can provide relatively high bandwidth (300 Kbps to 1.2 Mbps) over limited ranges (10 to 50 meters) by exploiting the unlicensed frequency band around 2.4 GHz. Table 1 gives the specifications for one of these technologies, Bluetooth, while Table 2 gives the specifications for the other, HomeRF.

While these two radio frequency (RF) technologies compete for a similar market, no matter which one prevails, future portable and embedded computing devices will clearly come equipped with some RF technology that will permit relatively high speed communications over a restricted range. Such technology will provide the communications conduit for large numbers of transient devices to begin to talk. But how will these devices discover each other and establish links and what will the devices say to one another? That's where Java and Jini come into the picture.

Most people have heard about the Java promise of write-once, run-anywhere software. Java makes this possible by requiring Java-compliant systems to implement an interpreter for a Java Virtual Machine (Java VM). In fact, a number of initiatives are underway to design chips that run the Java VM in hardware. Additionally, Sun and JavaSoft have been designing an operating system, JavaOS, intended to provide operating services native to Java programs by implementing a small kernel around a JavaChip. No matter what the outcome of these explorations, it appears possible to implement the Java VM on a chip (Java or otherwise) with fairly substantial memory and reasonable speed. Combining a Java VM-on-a-chip with a RF transceiver-on-a-chip could provide an interesting basis for networking pervasive computing devices, especially with the advent of Jini.

Table 1. Planned Specifications for Bluetooth Version 1.0	
• Range 10 Meters in shirt pocket or briefcase	• Point-to-point TCI/IP support
• Network Size: 8 devices per pico-net	• Low power standby mode
• Data Rate: 300-400 Kbps	• Higher transmit power possible
• Frequency: 2.4 GHz and 1600 Hops/sec	• Based on a working prototype
• Supports 3 near line-quality voice links	• More: http://www.bluetooth.com
• Optimized for cell phones and mobile devices	• Main players: Ericsson, IBM, Intel, Nokia, and Toshiba
• Multi-point to point connections	

Table 2. Planned Specifications for HomeRF Version 1.0	
• Range: 50 Meters in home and yard	• Native TCI/IP support
• Network Size: unlimited	• Low power paging mode
• Data Rate: 1.2 Mbps	• Lower transmit power possible
• Frequency: 2.4 GHz and 50 Hops/sec	• Based on shipping 802.11 and DECT technology
• Supports 6 near line-quality voice links	• More: http://www.homerf.org
• Optimized for home voice and data	• Main players: AMD, Ericsson, HP, IBM, Intel, Microsoft, Motorola, National Semiconductor, and more
• Peer-to-peer networking	

Jini is a Java-based networking technology recently announced by Sun. Jini enables devices, newly added to a network, to discover a lookup service and to deposit there some key information. This information can include a description of the device and its services, along with Java classes that can be used by others to communicate with the device. In addition, Jini enables programs and devices to discover other devices in an area, and then to download Java code that permits communication with the discovered devices. Along with Jini comes extensions to the Java VM to support event distribution among distributed Java VMs, and extensions to Java Remote Method Invocation (RMI) to exploit multicast protocols.

As should now be clear, the ingredients exist to provide pervasive computing devices with powerful networking functionality in a small, low-power package. Such a package would include a RF transceiver-on-a-chip, a hardware implementation of the Java VM, and enough memory to run the Jini discovery protocols, to hold Java classes for uploading to a Jini lookup service, and to execute Java classes downloaded from a Jini lookup service. Our proposed ***AirJava*** adapters would demonstrate the feasibility of these ideas, while at the same time providing a means to prototype tomorrow's pervasive computing devices today.

Planned Tasks

The birth of ***AirJava*** networking for pervasive computing requires us to complete three tasks.

Task 1. Design *AirJava* Adapter. Design of an ***AirJava*** adapter must cover both hardware and software. Hardware for a prototype ***AirJava*** adapter must provide a reasonably capable CPU, a significant amount of flash memory, a modest amount of DRAM, a serial port, a parallel port, a universal serial bus port, and the ability to accept PCMCIA interface cards, including either a RF or IR interface. The packaging must be reasonably compact, and power must be provided with batteries. To support development, the adapter should also accept PCMCIA interfaces for a secondary storage device, such as floppy disk. Software for the prototype ***AirJava*** adpater must provide an operating system or run-time environment capable of executing a Java Virtual Machine and running Jini.

Two approaches to design appear feasible. The conservative approach would base the ***AirJava*** adapter on a small, portable computer, such as the Toshiba Libretto. The Libretto provides a fully functional PC-like device in a reasonably compact form, powered with batteries. The Libretto can accept up to two PCMCIA cards. The Libretto can run Windows 95, Windows 98, or Windows NT, and thus can easily execute a Java Virtual Machine and Jini. On the downside, the Libretto is a bit heavy (just under 2 pounds) and has a short battery life (around 2 hours). As an alternative to the Libretto, we could consider some of the newer WindowsCE devices beginning to appear on the market. It is unclear whether WindowsCE will support the Java Virtual Machine; however, such devices are generally smaller and have longer battery life that computers in the Libretto class.

A less conservation approach would base the ***AirJava*** adapter on GUMPS, a GloMo Universal Module Packaging System developed for DARPA by USC-ISI. GUMPS provides a compact brick that includes a rigid-flex circuit board, a PCMCIA-format CPU, and up to seven PCMCIA cards. Four batteries, which can yield up to 10 hours of operation, power GUMPS. The GUMPS project also envisions a mini-brick (about the size of a pack of cigarettes) that would include a CPU plus up to five PCMCIA cards. The mini-brick would be powered on only two batteries, but an operating life of 20 hours is anticipated due to better power management. While the original plan for GUMPS called for a 33 MHz 486 CPU, as of 1998 a prototype GUMPS unit included by a 100 MHz 486DX4 CPU. The 1998 GUMPS prototype also included 150 Mbytes of flash storage. Producing a design based on a GUMPS-like architecture would enable us to consider a range of options for the CPU, including lower power RISC machines and JavaChips. We would also be free to consider software other than Windows. Options might include a free Unix variant, a real-time operating system, or JavaOS. Taking this approach would probably require some form of collaboration with USC-ISI.

Task 2. Produce Prototype *AirJava* Adapters. Once a design has been selected and verified, probably by implementing at least two working prototype ***AirJava*** adapters, we need to produce another ten adapters to support the concept demonstration planned for Task 3. In addition, we need to provide a cookbook of instructions so that others can produce their own ***AirJava*** adapters.

Task 3. Construct a Concept Demonstration of *AirJava* Networking. To show that the ***AirJava*** adapters work as expected, and to illustrate the concepts underlying networking for pervasive computing, we propose to demonstrate a small, pervasive computing environment. Our demonstration will include one Jini Server (connected to the global Internet as well as to a wireless cell), two portable computers, and a range of other devices. The range of other devices under consideration include a video projector, a videocassette recorder (VCR), a digital camera, a large-screen display, audio-visual equipment (such as found in the MASH room produced for DARPA by UCB), cell phones, the Cross Pad, wearable computer systems, and personal digital assistants. We will demonstrate that devices can be brought into a wireless cell, register with a lookup service, and be discovered by other devices and by individuals. We will also show that appropriate devices can download interface code for registered devices and then use that code to communicate with and control the registered devices. A canonical demonstration might have a person bring a video projector and VCR into a room, plug them into electrical outlets, and then leave. No other wires would be required. The devices would discover the lookup service and upload interface code. Later, a second person can enter the room with a portable computer. Using only the portable computer, the person will discover the interface code for the projector and VCR, download that code, pop in a video tape, remotely configure the VCR output to input to the projector, and then remotely control the playing of the tape, all from the portable computer. This will be accomplished without connecting the computer, VCR, and projector with cables. While the tape is playing, a third person can enter the room with a portable computer. This portable computer can also discover and download the interface code for the VCR and projector. Now, the new person can share control of the VCR with the existing person. Either person can pause, fast forward, play, and rewind the tape. Again, no communication wires.

Proposed Deliverables

Task 1	Design for ***AirJava*** Apater
Task 2	Twelve Prototype ***AirJava*** Adapters
Task 3	Demonstration of ***AirJava*** Networking for Pervasive Computing